AFRL-VA-WP-TP-2004-311

DECOMPOSITION TECHNIQUES FOR TEMPORAL RESOURCE ALLOCATION

Cynara C. Wu and David A. Castañon

APRIL 2004

Approved for public release; distribution is unlimited.

AIR VEHICLES DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7542

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*
April 2004

2. REPORT TYPE
Conference Paper Preprint

3. DATES COVERED *(From - To)*

4. TITLE AND SUBTITLE
DECOMPOSITION TECHNIQUES FOR TEMPORAL RESOURCE ALLOCATION

5a. CONTRACT NUMBER
F33615-01-C-3149

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
69199F

6. AUTHOR(S)
Cynara C. Wu (Alphatech, Inc.)
David A. Castañon (Boston University)

5d. PROJECT NUMBER
ARPF

5e. TASK NUMBER
04

5f. WORK UNIT NUMBER
T1

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Alphatech, Inc.
Burlington, MA 01803

Boston University
Department of Electrical and Computer Engineering
Boston, MA 02215

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Vehicles Directorate
Air Force Research Laboratory
Air Force Materiel Command
Wright-Patterson Air Force Base, OH 45433-7542

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/VACA

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-VA-WP-TP-2004-311

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
Conference paper to be presented at the 43rd IEEE Conference on Decision and Control (CDC), Paradise Island, Bahamas, 14-17 December 2004. This work, resulting from Department of Air Force contract number F33615-01-D-3149, has been submitted for publication in the Proceedings of the 2004 IEEE Conference on Decision and Control. If published, IEEE may assert copyright. If so, the United States has for itself and others acting on its behalf an unlimited, nonexclusive, irrevocable, paid-up, royalty-free worldwide license to use for its purposes.

14. ABSTRACT
We consider the problem of allocating a set of heterogeneous resources with availability constraints to maximize a given value function. The problem arises in a wide variety of military and industrial situations. We formulate the problem as a discrete-state decision process. We consider two instances of the problem that is applicable in situations where persistent coverage over all stages is desired. While we were able to solve the first example using dynamic programming, the computational requirements are significant and not scalable to larger instances. We consider an approximate technique using decomposition combined with dynamic programming. Our experiments show that this approach requires very little computation time and produces near-optimal results for the examples considered.

15. SUBJECT TERMS
Path Planning, Dynamic programming, Resource allocation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Mark Mears
Unclassified	Unclassified	Unclassified	SAR	12	19b. TELEPHONE NUMBER *(Include Area Code)* (937) 255-8685

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Decomposition Techniques for Temporal Resource Allocation

Cynara C. Wu and David A. Castañon

Abstract—**We consider the problem of allocating a set of heterogeneous resources with availability constraints to maximize a given value function. The problem arises in a wide variety of military and industrial situations. We formulate the problem as a discrete-state decision process. We consider two instances of the problem that is applicable in situations where persistent coverage over all stages is desired. While we were able to solve the first example using dynamic programming, the computational requirements are significant and not scalable to larger instances. We consider an approximate technique using decomposition combined with dynamic programming. Our experiments show that this approach requires very little computation time and produces near-optimal results for the examples considered.**

I. Introduction

THE allocation over time of heterogeneous resources with availability constraints is a problem that has applications in manufacturing, telecommunications, the military, and other areas. Examples include allocating different machines with required downtimes to jobs, as well as allocating vehicles with various capabilities and fuel limitations to different geographic areas to perform tasks. The majority of such problems are known to be NP-hard and numerous approximation techniques have been developed to solve them.

We consider an approximate solution based on decomposition. Decomposition approaches have been used to approximately solve a variety of hard combinatorial problems, including resource allocation and scheduling problems. They have been combined in a straightforward manner with sub-gradient and cutting-plane methods to solve simultaneous routing and resource allocation in wireless networks [3]. They have been combined with feature extraction for admission control in wireless networks [5]. They have been used to schedule test operations by decomposing the problem according to available work centers [4]. Work has also been done in developing novel heuristics for decomposing problems such as the Constraint Satisfaction Problem [2]. We present a straightforward approach based on decomposing the problem according to the available resources and solving the sub-problems using dynamic programming. While this approach has very likely been applied to combinatorial problems, we are unaware of specific applications based on dynamic programming for resource allocation problems.

The paper is organized as follows. In Section II, we formulate the problem. In Section III, we describe the dynamic programming solution to the problem. In Section IV, we describe the approximate solution based on decomposition. In Sections V and VI, we illustrate the solutions using two examples and present some computational results.

Manuscript received March 5, 2004. (Sponsor and financial support acknowledgment will go here).

C. Wu is with Alphatech, Inc, Burlington, MA 01803 USA; (phone: 781-273-3388; e-mail: cwu@alphatech.com).

D. Castañon is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA; (e-mail: dac@bu.edu).

II. Problem Formulation

We assume a set of R resources that are to be allocated to M groups over N stages. Resource i is available for up to t_i consecutive stages, at which point it is not available again until a minimum of s_i consecutive stages has passed. Fig. 1 provides a sample allocation schedule.

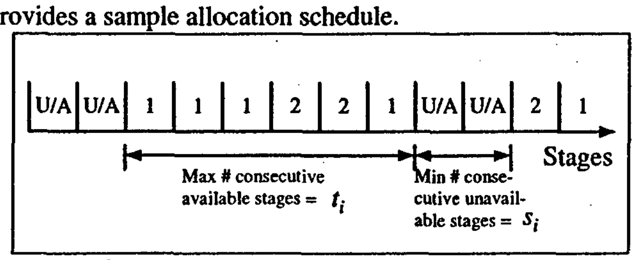

Figure 1 Allocation schedule where resource i is unavailable for 4 stages, and allocated to groups 1 and 2 for 5 and 3 stages, respectively.

The objective is to allocate resources subject to each resource's availability constraints to maximize a given value function. We assume that the value function accumulates over time; i.e., the value function is the sum of

A SC-04-0878

values attained during each stage and these values depend only on the states and decisions over each of the stages. We also assume that there is no cost in changing the allocation of one resource from one group to another between stages.

This problem can be formulated as a deterministic finite-horizon, discrete-state decision process. The state at stage k is represented by a vector $X_k = (x_{k1}, x_{k2}, \ldots, x_{kR})'$, where x_{ki} is an ordered pair that provides the availability and group information for resource i at stage k. Specifically, if resource i has been made available for stages k-n through k, was not available at stage k-n-1, and has been allocated to group m during stage k, then we have $x_{ki} = (x_{ki}^1, x_{ki}^2) = (n, m)$. In addition, if resource i has been made unavailable for stages k-n through k and was available at stage k-n-1, then we have $x_{ki} = (\min(0, n - s_i), 0)$. In the former case, the first element of vector x_{ki} is the number of stages for which resource i has been made available. In the latter case, the first element of vector x_{ki} is the number of stages until resource i will again be available.

The decision vector $U_k = (u_{k1}, u_{k2}, \ldots, u_{kR})'$ indicates which resources will be available to which groups during stage $k+1$. In particular, we have

$$u_{ki} = (u_{ki}^1, u_{ki}^2) = \begin{cases} (1, m) & \text{if resource } i \text{ is to be available to group } m \text{ at stage } k+1, \\ (0,0) & \text{otherwise.} \end{cases}$$

The possible choices for each decision component depend on the state at stage k. At any state, the decisions for any resource include making the resource unavailable for the next stage. For states where $x_{ki}^1 < t_i$, the decisions also include allocating the resource to any of the M groups for the next stage.

The state for each individual resource evolves as follows:

$$\begin{aligned} x_{(k+1)i} &= f_k(X_k, u_k) \\ &= \begin{cases} (x_{ki}^1 + 1, m) & \text{if } u_{ki} = (1, m),\ m \in \{1, 2, \ldots, M\}, \\ (-s_i + 1, 0) & \text{if } u_{ki} = 0 \text{ and } x_{ki}^2 \neq 0, \\ (x_{ki}^1 + 1, 0) & \text{if } u_{ki} = 0 \text{ and } x_{ki}^2 = 0. \end{cases} \end{aligned}$$

Given a set of value functions g_k, the optimal set of decisions is that which maximizes the value:

$$g_N(X_N) + \sum_{k=0}^{N-1} g_k(X_k, U_k).$$

Note that this problem can also be formulated as a general integer programming problem:

$$\max g(x)$$
$$\text{subject to } Ax \le b$$

where $x_{ijk} = 1$ when resource i has been allocated to group j during stage k and where A imposes the availability constraints for the resources. One possible approach to solving this problem is to enumerate all of the combinations of possible decision sets. However, the number of combinations is approximately $((M+1)^R)^N$. Although the dynamic programming solution described below is also exponential, its computational requirements are significantly smaller.

III. Dynamic Programming Solution

The decision process formulated in the previous section can be solved exactly using dynamic programming. (See [1] for a detailed treatment of this topic.) Let $J_k(X_k)$ denote the optimal value-to-go, or value that can be attained starting at state X_k at stage k. The value for every possible state for the last stage is simply $J_N(X_N) = g_N(X_N)$. Assuming we have the value-to-go for every possible state for the $(k+1)$st stage, we can determine the optimal decisions and the associated values-to-go for each possible state for the kth stage as follows. For each state, we consider each possible decision and compute the sum of the value attained over the current stage under the given decision and the value-to-go for the resulting state for stage $k+1$. The optimal decision and associated value-to-go for the state is that with the maximum sum:

$$J_k(X_k) = \max_{u_k} \{ g_k(X_k, U_k) + J_{k+1}(f_k(X_k, U_k)) \}.$$

At any stage, each resource can be in one of s_i unavailable states (since any state in which a resource has been made unavailable for more than s_i stages is equivalent to the state in which the resource has been made unavailable for exactly s_i stages), or in one of Mt_i available states. The number of states in this problem for a particular stage is therefore $\prod_{i=1}^{R}(Mt_i + s_i)$. For any state, the number of possible decisions is up to $(M+1)^R$, where the maximum number of decisions occurs when all resources can be made available for the next stage. The total number of decisions to evaluate to compute the optimal solution is then approximately

$$\left(\prod_{i=1}^{R}(Mt_i + s_i)\right)(M+1)^R N. \tag{1}$$

While complete enumeration is exponential in both the number of resources and the number of stages, this approach is only exponential in the number of resources. However, using dynamic programming to solve such problems exactly is still impractical for problems with a large number of resources, particularly if the problems need to be solved in real-time.

IV. Decomposition Approach

Since the computation time for obtaining an exact solution is exponential in the number of resources, a straightforward approach to addressing the computational limitations of exact dynamic programming is to decompose the problem into sub-problems associated with the individual resources. One application of this approach is to consider each resource in turn and select an allocation schedule for that resource by applying dynamic programming to determine the optimal solution for that individual resource assuming that the schedules for previously assigned resources are fixed and that there are no additional resources to allocate.

Let $O(i)$ provide the ordering in which the resources are to be considered. We first consider resource $O(1)$, assume that all remaining resources are never made available, and determine the allocation schedule $(x_{1O(1)}, x_{2O(1)}, \ldots, x_{NO(1)})$ that maximizes the value function $g_N(X_N)+\sum_{k=0}^{N-1} g_k(X_k,U_k)$. This allocation is then assigned to resource $O(1)$. For each subsequent resource $O(i)$, we assume that the schedules for resources $O(1), O(2), \ldots, O(i-1)$ are set to those previously computed and that the schedules for resources $O(i), O(i+1), \ldots, O(M)$ are never made available, and determine the allocation schedule $(x_{1O(i)}, x_{2O(i)}, \ldots, x_{NO(i)})$ that maximizes the value function $g_N(X_N)+\sum_{k=0}^{N-1} g_k(X_k,U_k)$. A general decomposition approach in which the sub-problems can be solved using any algorithm is illustrated in Fig. 1.

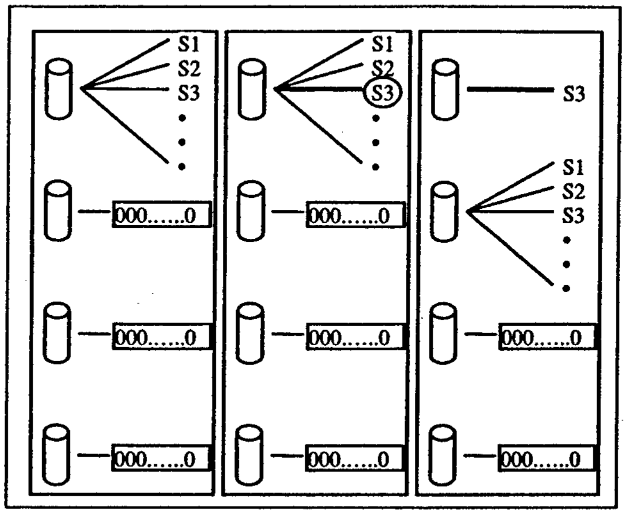

Figure 2 Illustration of a decomposition approach. Each resource, represented here as cylinders, is considered in some given order and allocated one of its feasible schedules. During the allocation of its schedule, two assumptions are made. First, remaining resources are allocated some pre-determined schedule, which in this example is one in which they are never allocated. Second, resources that have already been considered are fixed at the schedules previously determined. Any algorithm can be used to determine which of its feasible schedules to allocate to a resource. Our proposed approach uses dynamic programming to determine the optimal schedule given the two assumptions.

The number of states for the sub-problem associated with resource i is $Mt_i + s_i$. For each of these states, the number of possible decisions is either 1 or $M+1$. The number of decisions to evaluate to compute the optimal solution of the sub-problem associated with resource i is then bounded by $(Mt_i + s_i)(M+1)N$, resulting in the number of decisions to evaluate for the entire problem of

$$\sum_{i=1}^{R} \left[(Mt_i + s_i)(M+1)N\right]. \tag{2}$$

Since the allocations of resources that have not been assigned are not considered during the scheduling of a particular resource, the basic decomposition approach described above is a greedy algorithm. In fact, this approach is an example of an approximate dynamic programming algorithm where the value-to-go for unassigned resources is given an estimate of zero. As will be seen in the example described in the following section, this approach can obtain solutions that are very close to the optimal. However, for certain value functions, approximate dynamic programming approaches that provide more accurate estimates of the value-to-go are likely to obtain improved results. We are currently considering more complex value functions in order to evaluate additional approximate approaches.

V. EXAMPLE 1

The following example is motivated by a problem where the main objective is to ensure that surveillance over a single defined area of interest can be provided over an entire scenario.

A. Example Details

In this example, the objective is to allocate 23 vehicles over 36 stages to a single area of interest. The vehicles are of 4 different types, each of which has a surveillance range that is some fraction of the total area of interest. Vehicles carry a certain amount of fuel which allows them to patrol in the area of interest for a certain amount of time. When their fuel is low, they require a certain amount of downtime in order to travel to a station to refuel. The objective is to determine a schedule to indicate which vehicles are available during which stages that maximizes a given value function.

The following table provides for each vehicle type the number of available vehicles, the maximum number of consecutive stages each vehicle type is available before it needs to refuel, the minimum number of consecutive stages each vehicle type is unavailable after it heads to a station to refuel, and the fraction of the total area of interest for which each vehicle type can provide surveillance coverage.

Table 1 Resource Data for Example 1

Vehicle Type	Number	Max Availability	Min Off-Time	Coverage Provided
1	9	60	1	1/240
2	1	13	1	1
3	4	1	1	0.5
4	9	2	1	0.125

Since the objective of the problem is to provide sufficient surveillance at all times, we selected a value function that is independent of the stage and is monotonically non-decreasing in the total amount of coverage that could be provided by the vehicles allocated to a stage. In addition, the slope of the value function is greatest when the fraction of the coverage is less than 1 and decreases again when the fraction of coverage is greater than 1.5 and greater than 3. The decrease in slope as the fraction of coverage increases represents the decrease of importance of additional coverage when certain amounts of coverage have been reached. The function we selected is illustrated in the Fig. 2.

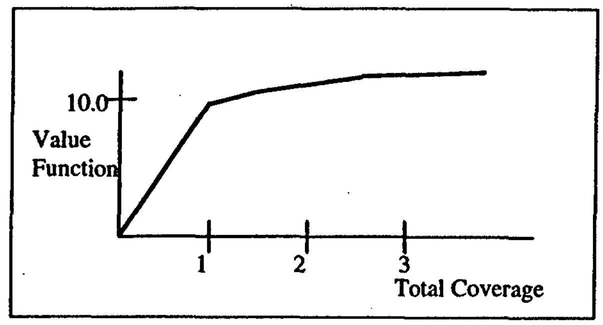

Figure 3 Value function for Example 1

B. Computational Results

Note that vehicles of type 1 can be made available for up to 60 stages. Since this is greater than the length of the scenario being considered and since the value function is monotonically non-decreasing in the amount of coverage available at any stage, any optimal solution would make these vehicles available during each stage. Note also that the vehicle of type 2 has such a large surveillance range that it can be shown that any optimal solution would limit the total number of unavailable stages to 2. Since the value function is uniform over all stages, it is straightforward to select a schedule for the vehicle of type 2 that would result in the optimal schedule. Therefore, finding the optimal solution for this example reduces to finding the optimal schedule for vehicles of type 3 and 4.

Using (1), we see that to determine the optimal allocation of vehicles of type 3 and 4 using dynamic programming, the number of decisions that need to be evaluated is

$$\left(\prod_{i=1}^{R}(t_i+s_i)\right)(2)^R N=\left(2^4\cdot 3^9\right)(2)^{13}(36)\approx 90.7\text{ billion.}$$

Using (2), we see that to determine the allocation of vehicles of type 3 and 4 using vehicle decomposition with dynamic programming, the number of decisions that need to be evaluated is

$$\sum_{i=1}^{R}[(t_i+s_i)\cdot 2]N=(9\cdot 3\cdot 2+4\cdot 2\cdot 2)\cdot 36=2520.$$

The results of applying the two approaches and their computational requirements are displayed in the following table. The algorithm was implemented in C on a Pentium 1.8 MHz processor. As can be seen, the decomposition approach obtains a schedule with a value that is within 0.1% of the optimal solution and requires approximately 6 orders of magnitude less computation time.

Table 2 Computational Results for Example 1

Solution Approach	Value	Number of Decisions	Computation Time
Exact DP	396.52	90.7 billion	68 minutes / 4080 sec
Decomposition	396.22	2520	0.002 seconds

VI. EXAMPLE 2

The next example is motivated by a problem where the main objective is to maintain persistent coverage over several geographic areas of interest to ensure that time-sensitive targets can be attacked within the required time window.

A. *Example Details*

In this example, the objective is to allocate 16 vehicles over 12 stages to 4 areas of responsibility in a manner that maximizes the expected value of targets that can be destroyed. The vehicles are of two types and are distinguished by the number of munitions they can carry. The munitions are of two types and are distinguished by their probabilities of destroying various targets. Table 3 provides the availability and munitions data for each vehicle type.

Table 3 Resource Data for Example 2

Vehicle Type	Number	Max Avail	Min Off-Time	Type 1 Weapons	Type 2 Weapons
1	8	5	3	4	2
2	8	5	3	2	1

The targets are also of two types: high value, time-sensitive targets that appear during each stage and can only be destroyed during that stage, and low value, non-time-sensitive targets that appear at the beginning of the first stage and remain until they are destroyed. The value attained during a particular stage was the expected value of destroyed targets if the allocated munitions were optimally assigned to the two types of targets.

Using (1), we see that to determine the number of decisions that need to be evaluated to obtain the dynamic programming solution is

$$(4\cdot 5+3)^{16}\cdot(5)^{16}\cdot 12=(115)^{16}\cdot 12\approx 1.12\cdot 10^{34},$$

which is too large to compute. We instead also formulated the problem as a general integer programming problem and attempted to use a commercial product to obtain the optimal solution. Unfortunately, it was also unable to compute the optimal solution after many hours of computation time.

Using (2), we see that to determine the number of decisions that need to be evaluated to obtain a solution using vehicle decomposition with dynamic programming is

$$[(4\cdot 5+3)\cdot 5\cdot 12]\cdot 16=22080.$$

At the time of this submission, we have implemented a vehicle decomposition approach using pruned enumeration instead of dynamic programming. This approach compares a subset of all possibilities to solve the vehicle sub-problems. The number of combinations considered was approximately 6000 for each vehicle, yielding a total of approximately 96000 total evaluations. The possibilities that were pruned from the entire search space included those where vehicles were not made available for extended periods of time and those in which vehicles were allowed to move from one area of responsibility to another during a particular availability period. We expect that the solutions for the sub-problems using dynamic programming will not vary much from those obtained under this approach.

The vehicle decomposition approach was able to obtain an approximate solution in 1.3 seconds that yielded a higher expected value than any solution obtained using the commercial product CPLEX after several hours. The value obtained as a function of time is illustrated in the following figure. We expect that the vehicle decomposition approach with dynamic programming will yield similar values using less computation time.

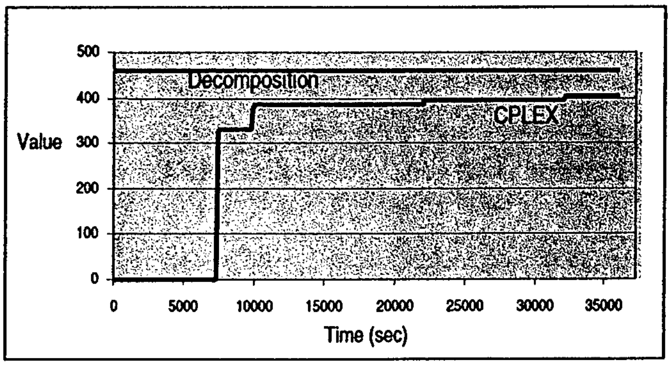

Figure 4 Value attained as a function of time.

VII. CONCLUSION

We have formulated the problem of allocating heterogeneous resources with availability constraints over time and space as a finite-state decision process and proposed a decomposition approach using dynamic programming as a solution. Our approach can be viewed as an approximate dynamic programming technique where the value that could be attained from unassigned resources is set to 0. The examples provided show that this approach is computationally feasible for some fairly large problems and that it obtains solutions that are very close to the optimal for some simple value functions. Ongoing work includes extensions involving more complex value functions that may require assigning approximate values to unassigned resources in order to obtain high quality solutions.

REFERENCES

[1] D.P. Bertsekas, *Dynamic Programming and Optimal Control*, Vol. One, 2nd Edition, Athena Scientific, 2000.

[2] B.Y. Choueiry and B. Faltings, "A Decomposition Heuristic for Resource Allocation," *11th European Conference on Artificial Intelligence*, 1994.

[3] L. Xiao, M. Johansson, and S. Boyd, "Simultaneous Routing and Resource Allocation via Dual Decomposition", to appear in *IEEE Transactions on Communications*, 2004.

[4] C.C. Wu and D.P. Bersekas, "Admission Control for Wireless Networks," Lab. For Info. And Decision Systems Report LIDS-P-2466, Massachusetts Institute of Technology, 1999.

[5] W.S. Yoo and L.A. Martin-Vega, "A Decomposition Methodology for Scheduling Semiconductor Test Operators for Number of Tardy Job Measures," *Journal of Electronics Manufacturing*, Vol. 7, No. 1, 51-61, 1997.